DOKUZ EYLUL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME1252 PROJECT BASED LEARNING – II
FINAL REPORT
PROJECT – I

by

Eray Işık

Beyza Gümüş

Görkem Dal

Güneş Zorgül Yıldırımlar

Mehmet Emir Koç

Lecturers

Assoc. Prof. Semih UTKU

Dr. Mustafa ERŞAHİN

Res. Assist. Onur Can DOĞANLAR

05.01.2024

CHAPTER ONE

PROGRESS DESCRIPTION

The primary objective of "The Cube (Robothon)" project was to develop an engaging game software featuring robot contests as its focal point. Throughout the project, participants were involved in a range of activities, from generating randomized pieces for human players to designing and refining robots tailored for different challenges. The pinnacle of the project was the RoboGames, where these crafted robots competed head-to-head, providing an exhilarating experience for all involved. This endeavor showcased both ingenuity and technical expertise in game software development, fostering significant learning opportunities throughout its duration.

CHAPTER TWO

TASK SUMMARY

2.1 Completed Tasks

Eray Işık : I took part in the development of the screen. I also created the constructors inside the classes and I was able to access the data inside by using the get methods. At the same time, I was involved in reversing the pieces and wrote the algorithms for RoboGames.

Beyza Gümüş: I have designed the entire code structure for the game. I created the project with classes such as Cube, Piece, Robot, GameScreen, EndScreen, Game, and Main. I developed the internal code for these classes.

I created the cubes and implemented cube validation using two boolean conditions (isEmpty and isNull).

I created the pieces within the Piece class, containing all the necessary properties for the pieces.

I implemented the Robot class to accommodate two types of robots. I calculated the robot forces within the methods of this class.

I generated 20 random pieces for the human robot and displayed them on the screen.I created a screen for designing the human robot.

I coded the piece selection with the arrow keys. I coded the Piece rotating and reversing (3,4,5 keys) operations.

I implemented the calculation of the average values for rows and columns within the pieces.

I programmed the movement of the matrix cursor within the cubes using the WASD keys.

I implemented the human robot design functionality on the screen.

I implemented piece placement and erasing functionality using the 1 and 2 keys.

I programmed the information area to dynamically update with the data and displayed it on the screen.

Additionally, I included a login screen to welcome the player at the start of the game. (The game starts upon pressing the 6 key)

Görkem Dal: Creating cubes and placing them on the start screen with random shapes and randomly placing numbers into these cubes and calculating the average score of the cubes with these numbers and printing it on the screen

Güneş Zorgül Yıldırımlar: I completed all the necessary tasks.First I created a cube class, a piece class from cube arrays. Computer robots class to calculate the skill, intelligence and the speed of the robot. I added reverse rotate and shift functions and implemented them to 3,4,5 keys. I added the ability of choosing the piece and choosing the robot matrix cursors place. According to the cursor places I added an placement ability to place the pieces cubes to the humans robots cubes. I colored the ending screen and shortened it.

Mehmet Emir Koç : Reversing the cubes on the piece depot section and creation of the end screen(RoboGames). Creation of the Cube class and necessary functions and loops.

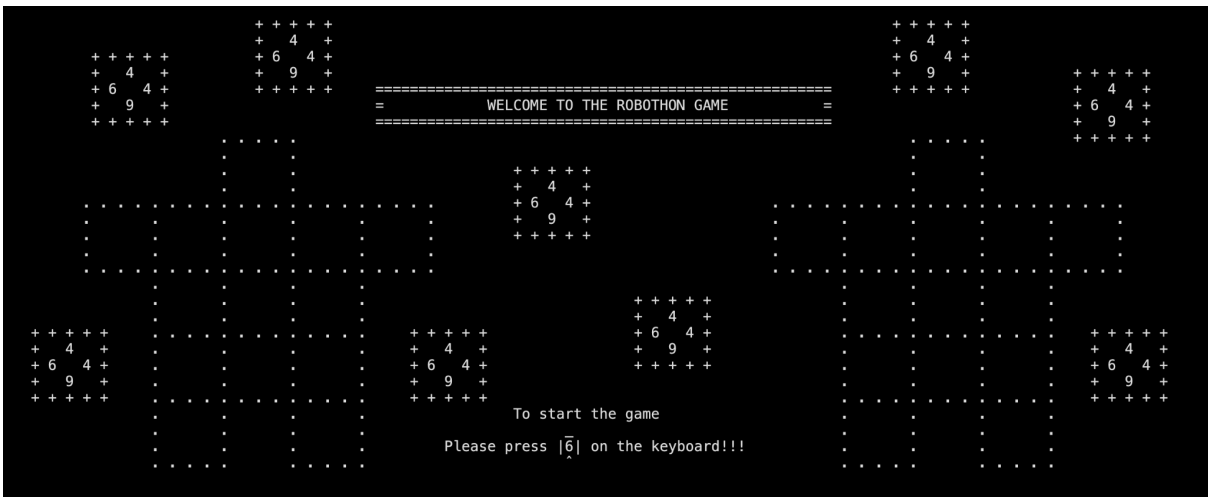
2.2 Incomplete Tasks: Reasons and Explanations

There aren’t any incompleted tasks.

2.3 Additional Improvements to the Project

Eray Işık :

Beyza Gümüş: Prior to game commencement, I crafted a welcoming login screen for players, complete with functionality to transition to the game interface upon pressing the '6' key.



Login screen

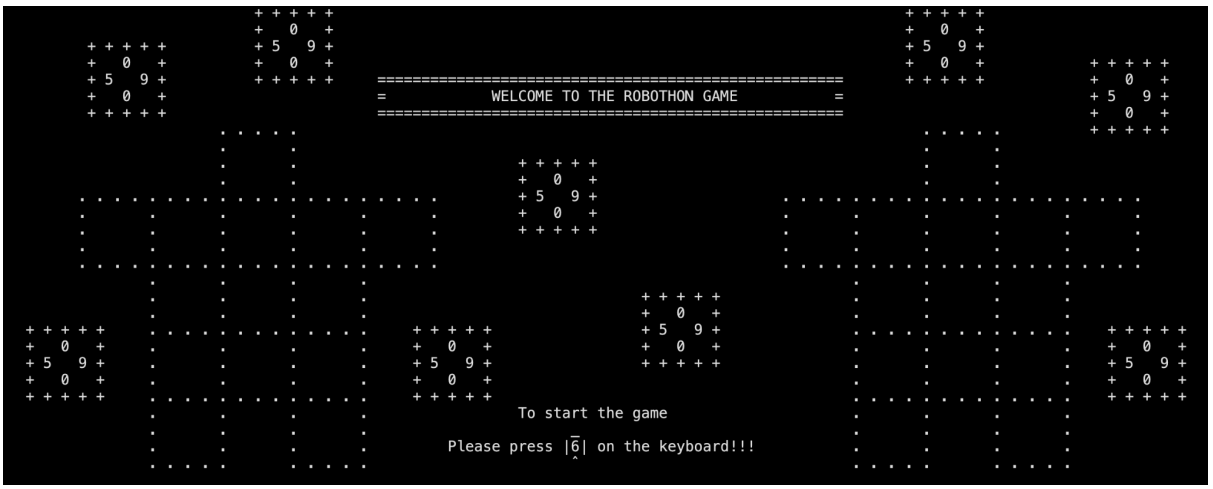
Görkem Dal:

Güneş Zorgül Yıldırımlar: I colored the end screen to make it look more appealing.

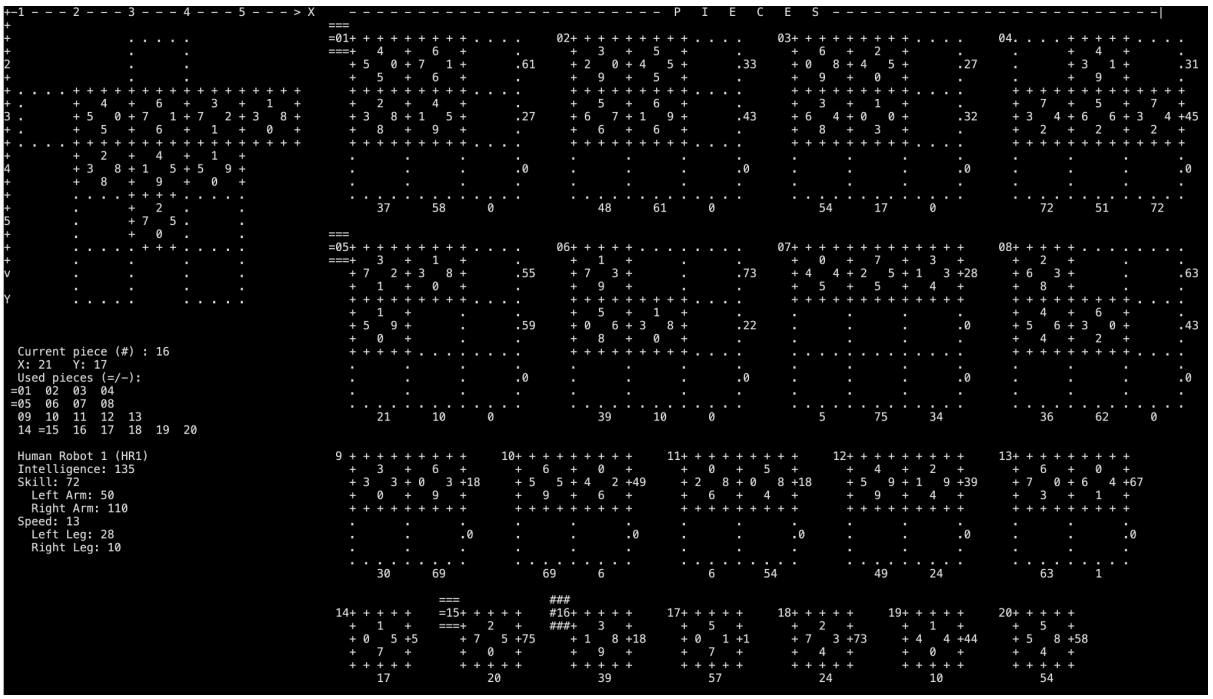
Mehmet Emir Koç :

CHAPTER THREE
EXPLANATION OF ALGORITHMS

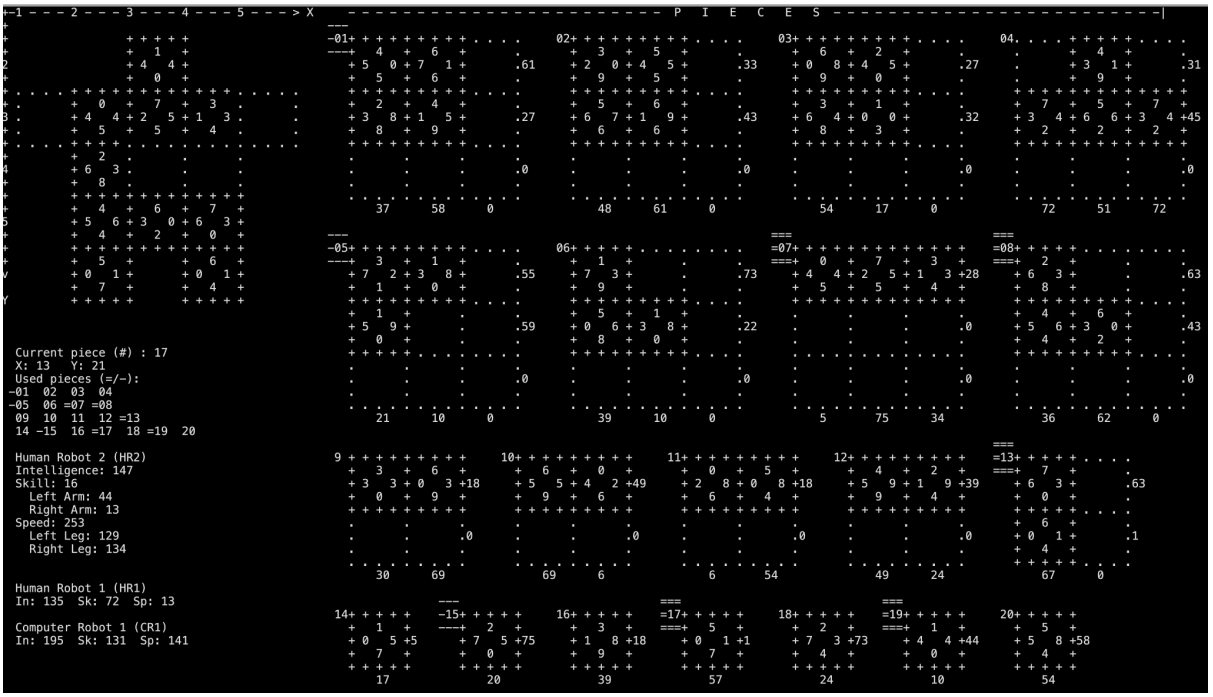
3.1 Screenshots



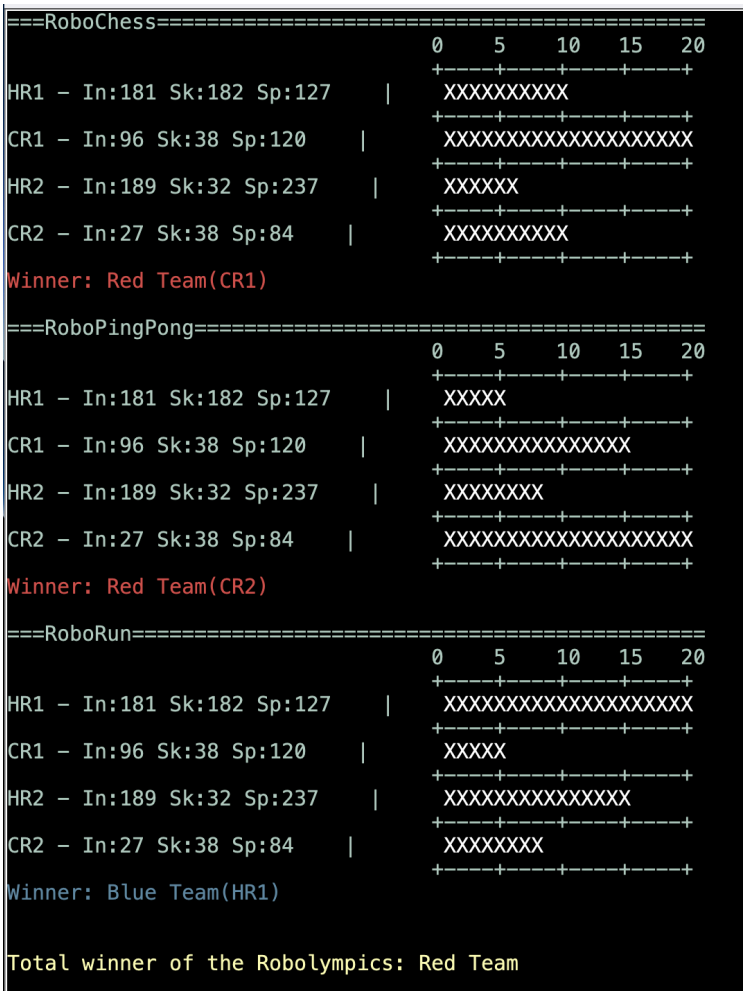
Login Screen



Designing screen for the first human robot

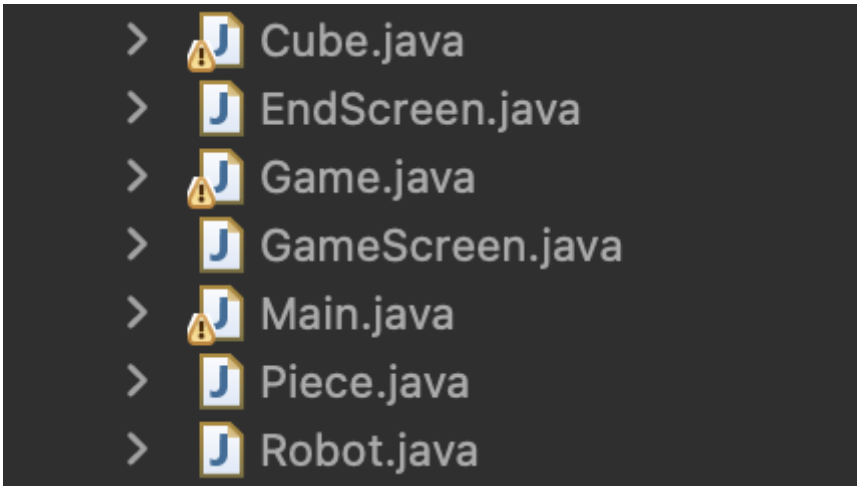


Designing screen for the second human robot



End screen of the game

3.2 Functions



All classes within the Robothon Game

3.3 Algorithms and Solution Strategies

Eray Işık : In the part of creating the cubes, I used x force and y force values and created random numbers for these values because we needed these values when calculating the averages of the cubes.

I mostly worked with the x values during the reversing process because when reversing the cube, the positions of the x values were completely rearranged. Of course, I saved their values when changing the x-forces and performed the necessary reversing operations. After adjusting the x-forces, I completed the reversing process by making changes in the y-forces.

We created the robogames section together with Emir.We used the get method a lot in this section because we needed the robot's intelligent speed and skill data to establish the necessary algorithms. After receiving this data,We made the game playable by using the random class and conditional statements.

Beyza Gümüş: The Game class is where we play the game. It contains the key controls needed to play, which are set up using KeyListener. Additionally, within the class, we create an object of the GameScreen class named 'screen'. This allows us to access the methods and properties of the GameScreen object.

```java
public class Game {

    TextWindow console = Enigma.getConsole("CubeConsole").getTextWindow();


    public KeyListener klis;

    //Standard variables for keyboard

    public int keypr; // key pressed?
    public int rkey; // key (for press/release)

    private GameScreen screen = new GameScreen();

    Game() throws Exception { // --- Constructor


}
```

Game Class

```java
    }
    if (rkey == KeyEvent.VK_1) {
        Piece piece = pieces[pieceNumber];
        screen.getHumanRobot().addPiece(px, py, piece);
        screen.printRobot();
        screen.printInformationArea(px, py, pieceNumber);
    }
    if (rkey == KeyEvent.VK_2) {
        int pieceToRemove = screen.getHumanRobot().removePiece(px, py);
        if (pieceToRemove != 0) {
            pieceToRemove--;
            pieces[pieceToRemove].setUsed(false);
        }
        screen.printRobot();
        pieces[pieceNumber].addMark();
        screen.printInformationArea(px, py, pieceNumber);
    }
    if (rkey == KeyEvent.VK_3) {
        pieces[pieceNumber].rotateLeft(pieces[pieceNumber].getCoordinateX(), pieces[pieceNumber].getCoordinateY());
    }
```

Sample usages of KeyListener(for 1,2,3 keys)

The GameScreen consists of components displayed during the game, such as robots, pieces, and information areas. This class manages the visual interactions of these components.

In the GameScreen class, an object named 'pieces' is created to access the 20-piece collection. Additionally, four different robots are instantiated using the Robot class. Notably, robots are created in two distinct manners: humanRobot and computerRobot. Coordinate values are also provided to enable the display of human robots on the GameScreen.

```java
public class GameScreen {

    Piece[] pieces = new Piece[20];
    Robot humanRobot1 = new Robot(2, 2);
    Robot humanRobot2 = new Robot(2, 2);
    Robot computerRobot1 = new Robot();
    Robot computerRobot2 = new Robot();


    TextWindow console = Enigma.getConsole("CubeConsole").getTextWindow();

    public GameScreen() {

    public Piece[] getPieces() {
        return pieces;
    }
```

GameScreen class

The Robot class utilizes the Cube class as a 2-dimensional array. The Robot class contains two different constructors. Since the human robot will be displayed on the screen, it is created with robot coordinates. However, the computer robot does not contain a coordinate. Additionally, a boolean variable named isCompleted is used to understand if the design of the first robot is completed.

```java
public class Robot {
    Cube[][] cubeMatrix;
    int x;
    int y;
    boolean isCompleted;
    TextWindow console = Enigma.getConsole("CubeConsole").getTextWindow();

    public Robot(int initialX, int initialY) {

    public Robot() {
```

Robot class

The operations of adding and removing parts to the robot are performed within this class. Additionally, score calculations for the Robot are also done in this class.

To track pieces during addition and removal and prevent them from getting lost on the robot, I assign a piece number to each cube. When removing a piece, I locate and remove the cube with that piece number from the robotMatrix.

```java
public void addPiece(int px, int py, Piece piece) {
    int xIndex = (px - x) / 8;
    int yIndex = (py - y) / 4;

    if (!piece.isUsed()) {
        Cube[][] pieceCubes = piece.getCubes();
        int pieceLength = pieceCubes.length;
        int robotLength = cubeMatrix.length;
        boolean failed = false;
        for (int i = 0; i < pieceLength; i++) {
            for (int j = 0; j < pieceLength; j++) {
                Cube pieceCube = pieceCubes[i][j];
                if (!pieceCube.isEmpty() && (i + yIndex < robotLength && j + xIndex < robotLength)) {
                    Cube robotCube = cubeMatrix[i + yIndex][j + xIndex];
                    if (!robotCube.isEmpty() || robotCube.isNull()) {
                        failed = true;
                    }
                }
            }
        }
        if (!failed) {
            for (int i = 0; i < pieceLength; i++) {
                for (int j = 0; j < pieceLength; j++) {
                    Cube pieceCube = pieceCubes[i][j];
                    if (!pieceCube.isEmpty() && (i + yIndex < robotLength && j + xIndex < robotLength)) {
                        cubeMatrix[i + yIndex][j + xIndex] = pieceCube;
                        cubeMatrix[i + yIndex][j + xIndex].setPieceNumber(piece.getPieceNumber());
                    }
                }
                piece.setUsed(true);
            }
```

Piece Adding Operation

To find the cube on the robot corresponding to the # symbol on the screen, I calculate its position based on the robot's initial position. I calculate the related length based on cubes, avoiding the use of hardcoded coordinates.

int xIndex = (px - x) / 8;: This line calculates the index of the cube in the x-direction. It subtracts the initial x-coordinate of the robot (x) from the current x-coordinate of '#' (px) and divides it by 8, which is the length of each cube in the x-direction.

```java
public int removePiece(int px, int py) {
    int xIndex = (px - x) / 8;
    int yIndex = (py - y) / 4;
    Cube robotCube = cubeMatrix[yIndex][xIndex];
    int pieceNumber = robotCube.getPieceNumber();

    for (int i = 0; i < cubeMatrix.length; i++) {
        for (int j = 0; j < cubeMatrix.length; j++) {
            Cube curentRobotCube = cubeMatrix[i][j];
            if (!curentRobotCube.isEmpty() && !curentRobotCube.isNull() && curentRobotCube.getPieceNumber() == pieceNumber) {
                cubeMatrix[i][j] = new Cube(true);
            }
        }
    }
    return pieceNumber;
}
```

Piece Removing Operation

The Piece class utilizes the Cube class as a 2-dimensional array with a dynamic cube count. Piece sizes are determined, and data placement based on the cube count is facilitated using a switch-case structure. Coordinates and Piece numbers are maintained within the Piece. Additionally, the usage status of the piece on the robot is determined by the isUsed and isPreviouslyUsed attributes.

```java
public class Piece {

    private Cube[][] cubes;
    private int cubeCount;
    private int coordinateX;
    private int coordinateY;
    private int pieceNumber;
    private boolean isUsed;
    private boolean isPreviouslyUsed;
    TextWindow console = Enigma.getConsole("CubeConsole").getTextWindow();

    public Piece(int cubeCount, int pieceNumber) {
```

Piece class

A Cube can be created as either filled or empty, serving as a component of other classes. The isEmpty and isNull variables are used to check the cube's status. Additionally, each Cube keeps track of the number of the Piece it belongs to, facilitating the tracking of Cubes during piece addition and removal.

```java
public class Cube {

    private int xForce;
    private int yForce;
    private boolean isEmpty;
    private boolean isNull;
    private int pieceNumber;
    TextWindow console = Enigma.getConsole("CubeConsole").getTextWindow();

//first value of isEmpty is true
    public Cube(boolean isEmpty) {
```

Cube class

Görkem Dal: With the cubes formed in the cube class, certain shapes are created using 3x3 and 2x2 arrays, then I take these arrays and print them on the necessary parts of the screen.

I used the "#" symbol to place the parts on the robot accurately, ensuring they were positioned correctly according to their designated locations. This helped me assemble the robot with precision and efficiency, avoiding unnecessary complications.

When printing the averages of the cubes, I assigned certain fixed values to set their positions and at that position I took the powers of the neighboring cubes and printed the average accordingly.

Güneş Zorgül Yıldırımlar: I created a Cube class and used two constructors in this Cube class. This allowed me to store both empty cubes and filled cubes in a cube array in the future, and it also provided the freedom to use null as a third option. After creating the Cube class, I thought it was straightforward, so I wrote a class to define the properties of the computer robot.After that for the pieces, I created cube arrays with sizes dependent on the dimensions of the pieces and randomly generated 20 of them. I added rotation and inversion features for these pieces. Since the pieces need to stay in the top left corner, I wrote a shift function. I printed the basic components of the screen.

While printing the pieces, I stored their positions in a two-dimensional array. With this position array, I easily moved the cursor.

```
int[][] pieceplaces = { { 1, 47 }, { 1, 77 }, { 1, 107 }, { 1, 137 }, { 16, 47 }, { 16, 77 }, { 16, 107 },
        { 16, 137 }, { 31, 47 }, { 31, 71 }, { 31, 95 }, { 31, 119 }, { 31, 143 }, { 42, 47 }, { 42, 61 },
        { 42, 75 }, { 42, 89 }, { 42, 103 }, { 42, 117 }, { 42, 131 } };// pieces locations
```

I did the same for the robot matrix cursor.

```
int[][] robotpieceplaces = { { 2, 2 }, { 10, 2 }, { 18, 2 }, { 2, 6 }, { 10, 6 }, { 18, 6 }, { 26, 6 }, { 34, 6 },
        { 10, 10 }, { 18, 10 }, { 26, 10 }, { 10, 14 }, { 18, 14 }, { 26, 14 }, { 10, 18 }, { 26, 18 } };
```

To allow the user to create their robot, I created a 5x5 cube array and ensured that the selected piece was pasted to the specified location in the robot matrix using a key, I checked if the piece fits with a boolean. I colored the final screen and divided it into functions.

Mehmet Emir Koç : For the Cube class, I randomly assigned x and y-Forces to cubes between 1 and 75. For printing the cubes, I used for loops and for the numbers(forces) on the cubes, I used get and set functions which was created by our team members.

For the reversing, I changed the x values of the cubes on the piece matrix. For the calculating forces on a reversed cube, I assigned a cube's x-force to a temporary and after changing x and y forces with each other, I assigned y-Force to that temp value(initial x-Force).

We created RoboGames section(the endgame) with Eray. For the calculations of intelligence, speed and skill of the robots, we used basic get functions. For the games, we used basic random and if conditions for the team powers.

CHAPTER FOUR

PROBLEMS ENCOUNTERED

Eray Işık : While developing the algorithm of the Robogames section, we encountered the problem of not being able to accurately access the data of the human-created robot. While one of the robots was moving forward, the other was not making any progress. We solved this problem by taking advantage of Object Oriented Programming, using get and set methods more effectively, placing for loops in more appropriate positions, and also making the random part more stable so that the data flow works properly.We also encountered problems with screen sizes due to Enigma. We changed the screen sizes with our teammates to decide which size would work more efficiently and finally decided on the most suitable one.

Beyza Gümüş:

While traversing the matrix cursor (#) on the human robot, it was leaving blank spaces after the points it had already visited.

Consequently, when it reached the boundary of the cube, it would erase the point, resulting in visual glitches. To prevent this, I adjusted the calculation of the cursor coordinates so that it can only navigate within the cube. Since it starts from a fixed point, the incremented values ensure that the cursor never reaches the boundaries. This way, regardless of which key is pressed, the cursor always stays within the cube.

In the information area section, when the x and y coordinates change simultaneously with the cursor movement, it was not able to delete one digit from the previous value when new values were written. As a result, the value was incorrectly displayed on the screen. To prevent this, I added an extra space every time a digit is printed, resolving the issue.

Görkem Dal: While working on the project, I ran into a few problems. First, the parts on the screen weren't lining up properly when we printed them. Then, we had trouble getting the right averages when we turned the cubes. Lastly, when we tried moving parts from the human robot to the robot, they didn't go where they were supposed to, making it hard to store them.

Güneş Zorgül Yıldırımlar: At first I couldn't understand the basics of classes. After creating the cube class and then creating the pieces class with the cubes I understood the basics . Since I had a solid foundation for the rest of the project, I didn't struggle much once I allocated time for it.

Mehmet Emir Koç : While developing the end screen, we encountered an issue where the algorithm erroneously assigned all points from one team to a single robot due to errors in our code. For instance, HR1 would end up with all 20 points while HR2 had none. By adjusting the loop conditions and ensuring the correct implementation of our random function, we rectified this mistake. Additionally, when printing the averages, we faced screen shifting problems. This was resolved by adjusting our console sizes, allowing us to now display the averages accurately and in the correct position.

CHAPTER FIVE

CONCLUSION

Conclusion

In summary, this project significantly improved our grasp of object-oriented programming and Java, a new language for many of us. We delved into classes, learning to effectively encapsulate data and behavior. Additionally, it expanded our technical skill set and introduced us to new concepts and tools. Collaboration was crucial, highlighting the importance of effective communication and teamwork. We appreciate our instructors' guidance, our team's dedication, and the support from others. This experience deepened our programming understanding and underscored the value of collaboration in problem-solving and project execution.

Technical Skills

For many of our team members, The Cube was our first Java project. In an interval of limited time, we had to expertise in both a new language, Java, and a new coding style, Object Oriented Programming. We learned class structures and how to use and code functions in them.

Social Skills

Similar to other PBL projects, collaboration was essential for "The Cube (Robothon)" project. As we familiarized ourselves with a new programming language and approach, effective communication was crucial for understanding our team members' code. This frequent and

efficient communication not only facilitated game development but also enhanced our teamwork and collaboration skills significantly, both in crafting the game and delivering presentations.

REFERENCES

https://www.codecademy.com/learn/learn-java-object-oriented-programming

https://github.com/topics/object-oriented-programming

https://github.com/talhanai/public--OOP-Spring21,

https://docs.oracle.com/javase/8/docs/api/java/awt/Color.html

APPENDIX A POSTER/WEB PAGE OF THE PROJECT